Results on the Fitness and Population based Fault Tolerant Approaches using a Reconfigurable Electronic Device

Didier Keymeulen Adrian Stoica Ricardo Zebulum Vu Duong

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
didier.keymeulen@jpl.nasa.gov

Abstract. The paper presents and compares two approaches to design fault-tolerant evolvable hardware: one based on the fitness definition and the other based on the population statistics. The fitness approach defines, in an *explicit* way, the faults that the component may encounter during its life time and evaluates the average behavior of the individuals. The population approach uses the *implicit* information of the population statistics accumulated by the genetic algorithm over many generations. The paper presents experiments done using both approaches on a fine-grained CMOS Field Programmable Transistor Array (FPTA) architecture for the synthesis of a fault-tolerant XNOR digital circuit. Experiments show that the evolutionary algorithm is able to find a fault-tolerant design for the XNOR function that can recover functionality when lost due to not a-priori known faults, by finding new circuits configurations that circumvent the faults. Our preliminary experiments show that the population approach designs a fault-tolerant circuit with a better performance and in less computation than the fitness based approach.

1. Introduction

Long-term survivability of space systems, as required for example by outer solar system exploration and missions to comets and planets with severe environmental conditions, has recently been approached with new ideas, such as the use of biology-inspired mechanisms for hardware adaptation. The application of evolution-inspired formalisms to hardware design and self-configuration lead to the concept of evolvable hardware (EHW). In the narrow sense EHW refers to self-reconfiguration of electronic hardware by evolutionary/genetic reconfiguration mechanisms. In a broader sense EHW refers to various forms of hardware, from sensors and antennas to complete evolvable space systems that could adapt to changing environments and, moreover, increase their performance during the mission.

EHW can bring one main benefit to spacecraft survivability by preserving existing functions, in conditions where hardware is subject to faults, aging, temperature drifts and radiation, etc. The fault-tolerant property is extremely important for electronic components used in the space and nuclear industry where the components are continuously subjected to ion radiation. As the limits of VLSI technology are pushed towards sub-micron levels in order to achieve higher levels of integration, devices become more vulnerable to radiation induced errors. These radiation induced erros can lead to system failure. One of the goals of the future electonics is to design radiation immune electronic components [20].

We propose to produce electronic systems that are inherently insensitive to faults such as silicon defects by using evolution in hardware to design fault-tolerant or highly reliable systems. The evolution is even able to on-line self-repair by changing the circuit configuration in short delay or off-line self-repair by pushing further the evolution and exploiting defective components as if they were working parts [15] [16]. This paper reports on experiments that illustrate how evolutionary algorithms, using two different approaches, can design fault-tolerant digital circuit and recover functionality when lost due to faults, by finding new circuit configurations that circumvent the faults immediately in hardware.

A variety of circuits have been synthesized through evolutionary means. For example, Koza used Genetic Programming (GP) to grow an "embryonic" circuit to one that satisfies desired requirements [1]. This approach was used and extended for evolving a variety of circuits, including filters and computational circuits [2]. On-chip evolution was demonstrated for the first time by Higuchi [27]. Later Thompson [3] used an FPGA as the programmable device, and a Genetic Algorithm (GA) as the evolutionary mechanism and Kajitani [26] used a dedicated hardware integrating the GA computation and a reconfigrable hardware. More details on current work in evolvable hardware are found in [4], [5], [6], [7], [24]. More recently, evolutionary experiments were performed on Field Programmable Analog Arrays [18] and custom-designed ASIC [11][25]. Evolutionary algorithms have also been used with success for designing fault-tolerant system, such as robotics [15][21] and recently also in electronics [16][23][22].

This paper is organized as follows: Section 2 presents the fault tolerant principles and the evolutionary method to obtain fault-tolerant systems. Section 3 presents the FPTA concept and the experimental setup. Section 4 describes the fault tolerant experiments using a cascaded

FPTAs to design a XNOR logical function. Section 5 presents some lessons learned from the experiments and section 6 concludes the paper.

2. Fault Tolerant Principles for Evolvable Hardware

The definition of fault tolerance is simply that a fault in a component does not cause the overall system to malfunction [14]. The malfunction is in general a loss of service that can be total or partial as for example on a computer network. The characteristic of fault tolerance is not absolute. The question is one of degree: how much tolerance to faults is required varies from application to application. In our electronic experiment, the malfunction is calculated by the mean square error between a desired output DC characteristic and the actual output.

Fault tolerant systems are evaluated by two criteria: their *reliability* and their *availability*. The reliability measures how long can the system operate before malfunctioning even in the presence of faulty components. The availability measures the expected proportion of time that the system will be available for use. In our experiment on electronic device, the reliability of the circuit is measured by evaluating the malfunction of the electronic device when injecting faults. The availability is measured by calculating the time needed by the evolution process to retrieve a satisfactory circuit design.

Two principles for designing fault-tolerant systems can be applied for evolutionary design: *redundancy* and *on-line repair*. The redundancy concept is well understood: if part of a system fails, there is an "extra" or spare" that is able to operate in the place of the failed component such that the operation of the system is uninterrupted. The on-line repair imposes that the system with a failed component should be made unavailable as less as possible while the system is in service. These two principles can be applied to fault-tolerant evolutionary design. First, redundancy is obtained by using a circuit with a large number of connections and elements (transistors). Second, the on-line repair is obtained by searching, in the population, for a correct circuit, or by running the GA during a limited number of generations.

Two different approaches were proposed to build fault tolerant system using evolutionary algorithms:

1. *Fitness Based Fault-Tolerant Design*: it consists of injecting during the evolutionary process, the faults known a-priori that may occur in the circuit during its life-time [19].

2. *Population Based Fault-Tolerant Design*: it consists of extracting from a population of evolved circuits, the individual which adequately performs a desired functionality in the presence of a fault and eventually continue the evolution to attain a performance equal to that before the fault occurred [16].

While in the population based approach no previous knowledge of the faults that may occur is assumed, the fitness based approach requires a-priori knowledge of the defects. We will show in this paper that the population fault-tolerant approach using the population statistics accumulated by the genetic algorithm performs better than the fitness fault-tolerant approach. In the following section we present the FPTA and the evolutionary platform on which we conduct the experiments. Then the e periments and their results are described.

3. Test Bed for FPTA

The idea of a programmable transistor array was introduced first in [11]. The FPTA cell is an array of transistors interconnected by programmable switches. The status of the switches (ON or OFF) determines a circuit topology and consequently a specific response. Thus, the topology can be considered as a function of switch states, and can be represented by a binary sequence, such as “1011…”, where by convention one can assign 1 to a switch turned ON and 0 to a switch turned OFF. The FPTA architecture allows the implementation of bigger circuits by cascading FPTA cells. To offer sufficient flexibility the module has all transistor terminals connected via switches to expansion terminals (except those connected to power and ground). Figure 1 illustrates an example of a FPTA cell consisting of 8 transistors and 24 programmable switches. In this example the transistors P1-P4 are PMOS and N5-N8 are NMOS. A test chip implementing the FPTA architecture was developed. The programmable switches were implemented with transistors, acting as simple T-gate switches. Each chip contains one FPTA module and was fabricated as a Tiny Chip through MOSIS, using 0.5-micron CMOS technology. The test board with four chips mounted on it is illustrated in Figure 3.

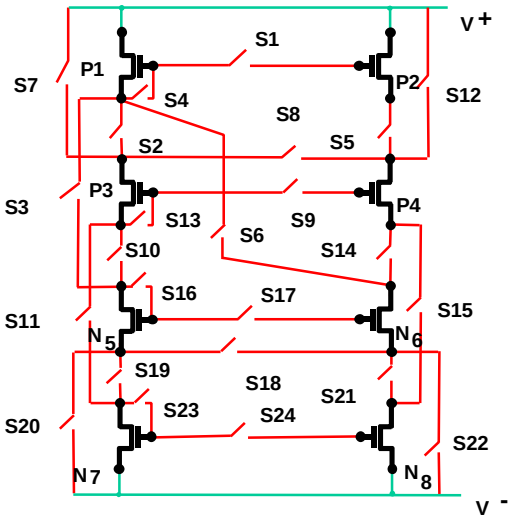

Figure 1. Module of the Field Programmable Transistor Array Cell

An evolutionary design tool was developed to facilitate experiments in hardware evolution [17]. The tool illustrated in Figure 2 uses the public domain Parallel Genetic Algorithm package, PGAPack and an evovable hardware test bed built around LabView. An interface code links the GA with the hardware where potential designs are evaluated, while a GUI allows easy problem formulation and visualization of results. At each generation the GA produces a new population of binary chromosomes, which get converted into configuration bits for the reconfigurable devices. Configuration bits are further downloaded into the hardware device by LabView. Circuit evolutionary synthesis directly on the chip became possible at an expected accelerated pace of over two orders of magnitude compared to the simulation on a workstation

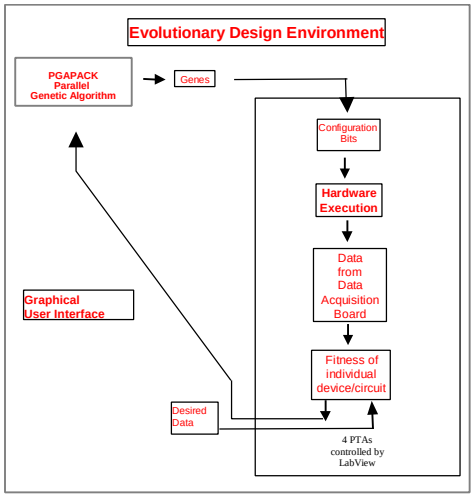

Figure 2. Environment for evolutionary hardware design.

4. Fault-Tolerant Experiments

The aim of this experiment was to test and compare the reliability and availability of a circuit design obtained by respectively a population and a fitness based evolution. The experiment setup consists of two cascaded FPTAs each programmed by 24 internal swiches. The 2 FPTA are connected together by 6 external wires controlled by 6 programmable switches (Figure 3). Each FPTA is connected through 4 programmable switches to two input voltages, one current bias and one output load. There are a total of 62 switches controlling the 2 cascaded FPTAs and representing the chromosome for the GA (Figure 4).

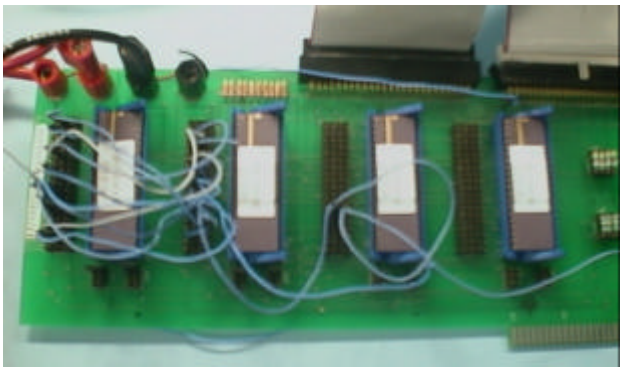

Figure . 3. A test board with 2 cascaded FPTAs (The two FPTAs used in the experiment are on the left side of the picture)

The experiment consisted of the evolutionary design of a XNOR logical function using two square wave voltage inputs, at frequency of 50Hz and 100Hz respectively (Figure 5). The fault tolerance test encompassed the introduction of six single faults on the external wires connecting the 2 FPTAs by imposing the switches to be ON (short fault) or OFF (cut fault). The evaluation function is the MSE between the output response of the circuit obtained by evolution and the ideal output signal of a XNOR logical function.

The experiments used a Genetic Algorithm (GA) with the following parameters: population 200, tournament selection of size 10; uniform mutation probability: 0.04, uniform cross-over probability: 0.7, elite strategy: 10%, fitness function: mean square error. The GA obtained the XNOR response in 60 generations, taking 3 minutes.

1. Population Fault-Tolerant Evolution

Evolution started by randomly initiating the population chromosomes, which were transformed into connection patterns. These are downloaded into the chips and the output of the generated circuits was directly monitored and compared with the desired DC XNOR response. After 60 generations a circuit that satisfied the requirements was found and is shown in Figure 6.

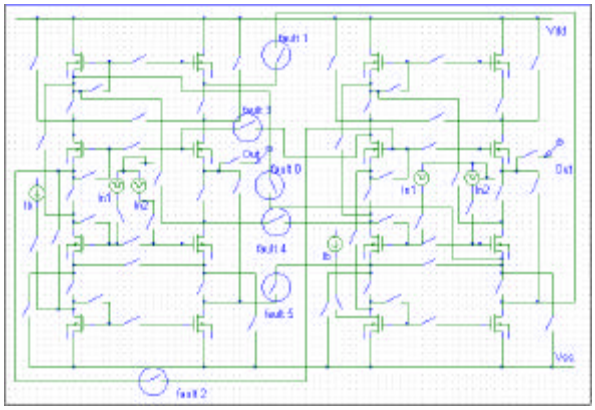

Figure . 4. Cascaded FPTAs used to design a fault-tolerant XNOR circuit with 62 switches. The 6 connections indicated by a circle are subjected to faults. (CUT: switch OFF: fault 0, fault 3, fault 5) (SHORT: switch ON: fault 1, fault 2, fault 4)

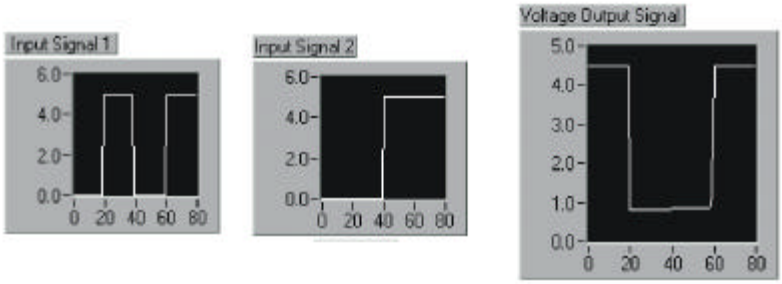

Figure 5. Input Signal 1(100Hz), Input Signal 2 (50Hz) and the Output Signal of the XNOR circuit configuration. (X axis: 0.25 msec/unit; switches: 1Volt/unit).

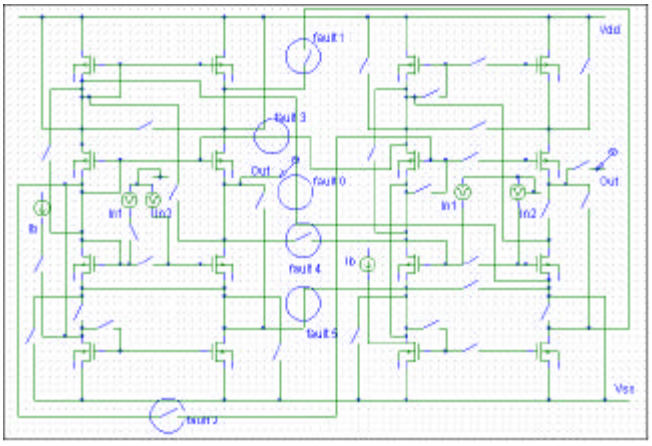

Figure. 6. Best Circuit Design at Generation 60

We inject five faults by cuting (set the switch OFF: fault0, fault3, fault5) or shorting (set the switch ON: fault1, fault2, fault4) one by one the external connections between the two FPTAs. Figure 7 shows that the best circuit configuration does not achieve the XNOR

functionality for faults 2, 3, 4, 5. Looking in the population at generation 60, we found mutants with better responses for faults 2 and 5 as shown in figure 8 and 10. However we could not find mutants with acceptable performance for fault 3 and 4 (figure 10). We then re-started the GA with the population of its last run evaluating the individuals under fault 3 and fault 4 conditions [15]. In case of fault 4, and starting with the last available population, it took half less generations (30 generations) to recover than when starting with a random population as shown on figure 9. In figure 10 we compare the performance of the best circuit and the mutants found for each fault. It illustrates that the population approach is able to find on-line a circuit configuration to resolve the faults.

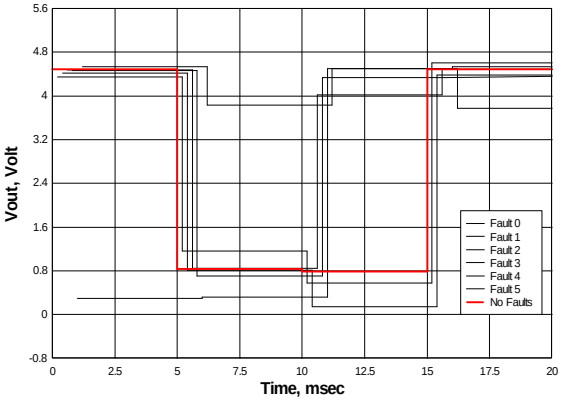

Figure. 7. Output of the Best Circuit Configuration obtained by population based evolution when 6 faults are injected. (The response are shifted in time in all the figures to enhance the illustration)

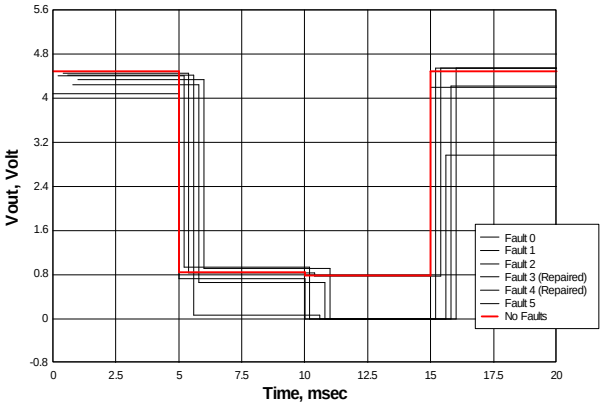

Figure. 8. Response of best performing mutants for each fault. Further evolution was needed to find a XNOR circuit for fault3 and fault 4.

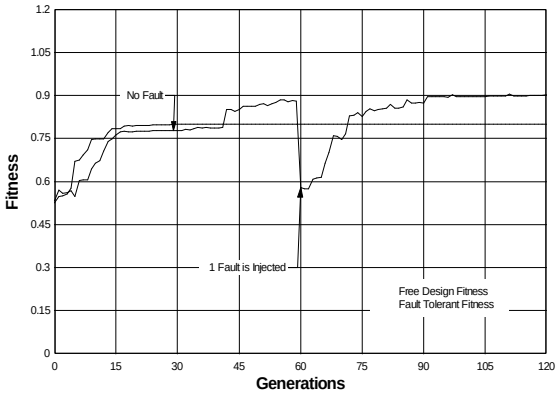

Figure 9. Fitness through generation for Population based and Fitness based fault-tolerant approach.

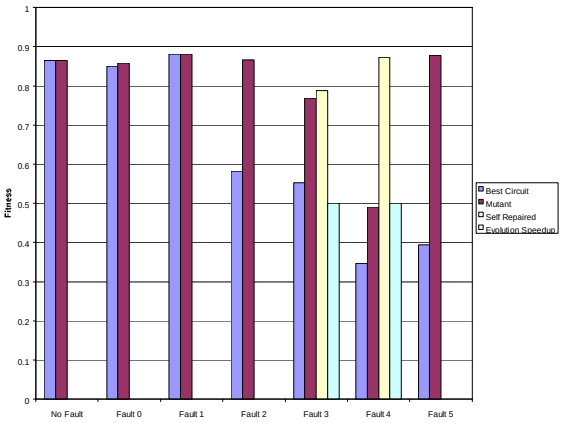

Fig. 10. Comparison between the Fitness of the Best Circuit and Mutant Circuits obtained by a population based evolution.

2. Fitness based Fault-Tolerant Evolution

In the fitness based fault-tolerant experiment, the chromosomes are evaluated in four different circuit states: one without fault, and three with a single fault. The three faults are fault 2 (switch ON), fault 3 (switch OFF) and fault 5 (switch OFF). The fitness of the chromosome is the average of the four evaluations. After 30 generations, the genetic algorithm finds a circuit that best satisfied the requirement (Figure 9). The circuit configuration is shown on Figure 11.

We then inject six faults. As expected the circuit achieves the XNOR functionality for each of the three faults included explicitly into the fitness function (fault 2, fault 3, fault 5) (Figure 12). The circuit is also able to achieve the XNOR functionality with faults not included into the fitness function such as fault 0, fault 1 and fault 4 but with a lower performance for fault 0 (Figure 13). Finally we applied inverse faults to the best circuit configuration. It shows that the circuit configuration cannot achieve the XNOR functionality when fault 2 is inverse.

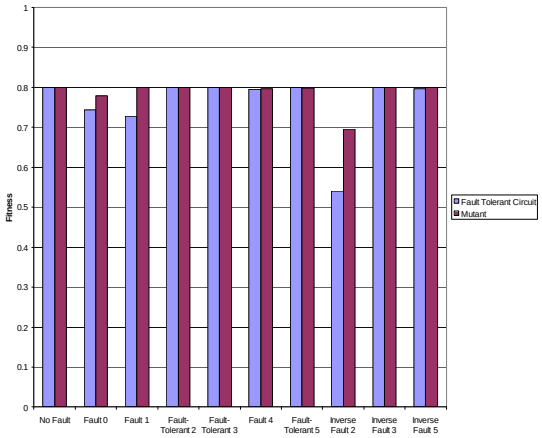

Figure 13. Comparison between the fitness of best and mutant configuration obtained by a fitness based evolution.

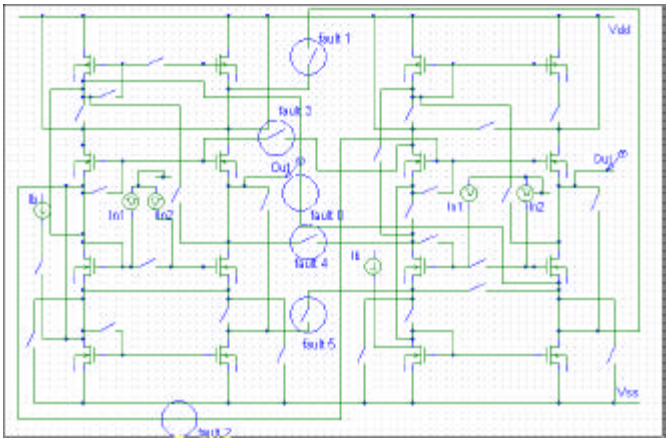

Figure 11. Circuit Schematic of the best individual obtained by fitness based approach.

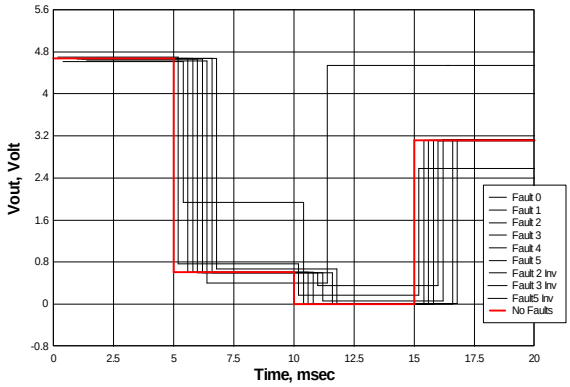

Figure 12. Output of the best circuit configuration obtained by fitness based evolution when 9 faults are injected

6. Lessons Learned

We compared two approaches for designing a fault-tolerant field programmable transistor array and we conclude from our preliminary experiments that the population approach offers the following advantages compared to the fitness approach.

- The population approach builds circuits with a better performance in a no-fault situation than the circuit obtained by the fitness approach, because, in the later case, the evolution is constrained by the faults imposed to the circuit. But the fitness approach has the advantage of achieving a single circuit robust to multiple faults.

- The population approach offers an on-line self-repair mechanism able to find circuit in the population with better performance than the circuits obtained by the fitness approach. Although the best circuit configuration for a non-fault situation is not robust, the population contains mutant configurations able to achieve the desire functionality with the faulty circuit. They even display a better performance than the best configuration and mutants obtained by the fitness approach.

- The population approach offers an off-line self-repair mechanism able to self-heal circuit in few more generations with better performance than the circuit obtained by the fitness approach.

- The population approach requires less computation than the fitness approach because in the later case the genetic algorithm must evaluate the circuits with the faults.

These experiments open the way for further investigation of the property of fault-tolerant evolutionary techniques applied to electronics such as the behavior of the fault-tolerant system when arbitrary and large number of fauts are injected and the unavaibility time is limited. The methodoly can also be addressed by combining the population and the fitness approaches, or by including in a more explicit way redundancy in the system such as explored in the "embryological" development approach [13].

8. Conclusion

The paper demonstrates the power of evolutionary algorithms to design digital fault-tolerant circuit. It compares two methods to achieve fault-tolerant design one based on fitness and the other based on population. It shows that although the classic fault-tolerant design approach is able to create a reliable circuit design by evaluating the behavior of the circuit when well known faults are injected during the evolutionary process, better circuit performance and in less computation time for a same fault-tolerant degree is achieved by allowing the evolutionary design process to be free of all faults constraints.

Acknowledgements

This research was performed at the Center for Integrated Space Microsystems, Jet Propulsion Laboratory, California Inst. of Technology and was sponsored by the Defense Advanced Research Projects Agency (DARPA) under the Adaptive Computing Systems Program.

Reference

[1] J. Koza, F.H. Bennett, D. Andre, and M.A Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming", *Proceedings of Genetic Programming Conference*, Stanford, CA , pp. 28-31, 1996

[2] J. Lohn, J. and S. Colombano, "Automated Analog Circuit Synthesis using a linear representation", M. Sipper, D. Mange and A. Perez-Uribe (Eds.*) Evolvable Systems: From Biology to Hardware*, Springer-Verlag Lecture Notes in Computer Science Berlin 1998, pp. 125-133

[3] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined in physics". In *International Conference on Evolvable Systems*. Springer-Verlag Lecture Notes in Computer Science, 1996, pp. 390-405.

[4] E. Sanchez and M. Tomassini (Eds.) *Towards Evolvable Hardware*, LNCS 1062, Springer-Verlag, 1996

[5] T. Higuchi, M. Iwata, and W. Liu (Eds.) Evolvable Systems: From Biology To Hardware, *Proc. of the First International Conference, ICES 96*, Tsukuba, Japan, Springer-Verlag Lecture Notes in Computer Science, 1997.

[6] M. Sipper, D. Mange, A. Perez-Uribe (Eds.) Evolvable Systems: From Biology To Hardware, *Proc. of the Second International Conference, ICES 98*, Lausanne, Switzerland, Springer-Verlag Lecture Notes in Computer Science, 1998.

[7] J. R. Koza, F. H. Bennett III,, D. Andre and M. A. Keane, Genetic Programming III – Darwinian Invention and Problem Solving, Morgan Kaufman, San Francisco, 1999

[8] E. Vitoz, Analog VLSI Processing: Why, Where and How, *Journal of VLSI Processing*, Kluwer, 1993

[11] Stoica, A. Toward evolvable hardware chips: experiments with a programmable transistor array. *Proceedings of 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, Granada, Spain, April 7-9, IEEE Comp Sci. Press, 1999.

[12] Layzell, P. A New Research tool for Intrinsic Hardware Evolution. In *Proceedings of ICES'98*, Lausanne, Switzerland, 1998

[13] P. Marchal et al. Embryological development on silicon. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 365-366. MIT Press, 1994.

[14] White R. and Miles F. Principles of Fault Tolerance. In *Proceedings of Eleventh Annual Applied Power electronic Conference and Exposition*, pages 18-25, Vol.1. IEEE Press, 1996.

[15] Thompson A. Evolving fault tolerant systems. In *Proceeding of the First Interntional Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, page 524-529. IEEE Press, 1995.

[16] Layzell, P. Inherent Qualities of Circuits Designed by Artificial Evolution: A preliminary study of populational fault tolerance. In *Proceedings of the First NASA/DoD Worshop on Evolvable Hardware*, pages 85-86. IEEE Computer Society Press, 1999.

[17] Stoica A., Keymeulen D., Tawel R., Salazar-Lazaro C., Li W. Evolutionary Experiments with a Fine-Grained Reconfigurable Architecture for Analog and Digital CMOS Circuits. In *Proceedings of the First NASA/DoD Worshop on Evolvable Hardware*, pages 76-84. IEEE Computer Society Press, 1999.

[18] Zebulum, R. et al., Analog Circuits Evolution in Extrinsic and Intrinsic Modes. In *Proc. of the Second International Conference, ICES 98*, Lausanne, Switzerland, Springer-Verlag Lecture Notes in Computer Science, 1998, pp 154-165

[19] Devarayanadurg G et al., Test Set Selection fro Structural Faults in Analog IC's. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp 1026-1039, Vol.18, No. 7, July 1999. IEEE Press.

[20] Niranjan, S. and Frenzel, J.F., A comparison of fault-tolerant state machine architecture for space-borne electronics. In *IEEE Transactions on Reliability*, pp 109-113, Vol.45, No. 1, March 1996. IEEE Press.

[21] D. Keymeulen, M. Iwata, Y. Kuniyoshi and T. Higuchi. On-line evolution for a self-adapting robotic navigation system using evolvable hardware. *Artificial Life*, 4(4):359-393, 1999. Special Issue on Evolutionary Robotics. MIT Press.

[22] Ortega C. and Tyrrell A. Reliability analysis of self-repairing bio-inspired cellular hardware. In *Proceedings of IEE Half-day Colloquium on Evolutionary Hardware Systems*, pp 2/1-2/5, 2 June 1999. IEEE Press.

[23] Zebulum, R et al. Evolvable Hardware: Automatic Synthesis of Analog Control Systems. In *IEEE Aerospace Conference*, Big Sky, Montana, March 14-25, 2000. IEEE Press .(submitted and approved)

[24] Higuchi T. et al. Real-World Applications of Analog and Digital Evolvable Hardware. In *IEEE Transactions on Evolutionary Computation*, Vol.3, No. 3, September 1999. IEEE Press.

[25] M. Murakawa, S. Yoshizawa, I. Kajitani, Xin Yao, N. Kajihara, M. Iwata and T. Higuchi The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing. In *IEEE Transactions on Computers*, vol. 48, no. 6, pp.628-639, 1999. IEEE Press.

[26] I. Kajitani, T. Hoshino, N. Kajihara, M. Iwata and T. Higuchi. An Evolvable Hardware Chip and Its Application as a Multi-Function Prosthetic Hand Controller. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pp. 182-187, 1999. AAAI Press.

[27] Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H., and Furuya, T. Evolving hardware with genetic learning: A first step towards building a Darwin machine. In Meyer, Jean-Arcady, Roitblat, Herbert L. and Wilson, Stewart W. (editors). *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. pp 417 - 424. 1993. Cambridge, MA: The MIT Press.